NUMERICAL RECIPES FOR MOLD FILLING SIMULATION

Doug Kothe[1], Damir Juric[2], Kin Lam[3], and Bryan Lally[4]

Los Alamos National Laboratory
Los Alamos, NM, 87545, USA

[1]Materials Science and Technology Division
Structure/Property Relations
Group MST-8, MS-G755
dbk@lanl.gov

[2]Theoretical Division
Fluid Dynamics
Group T-3, MS-B216
djuric@lanl.gov

[3]Engineering Sciences and Applications Division
Engineering Analysis
Group ESA-EA, MS-P946
klam@lanl.gov

[4]Materials Science and Technology Division
Structure/Property Relations
Group MST-8, MS-G755
lally@lanl.gov

Abstract

Has our ability to simulate the filling of a mold progressed to a point where an appropriate *numerical recipe* achieves the desired results? If "results" are defined to be topological robustness, computational efficiency, quantitative accuracy, and predictability, all within a computational domain that faithfully represents complex three-dimensional foundry molds, then the answer unfortunately remains *no.* Significant interfacial flow algorithm developments have occurred over the last decade, however, that could bring this answer closer to "maybe". These developments have been both evolutionary *and* revolutionary, will continue to transpire for the near future. Might they become useful numerical recipes for mold filling simulations? Quite possibly. Recent progress in algorithms for interface kinematics and dynamics, linear solution methods, computer science issues such as parallelization and object-oriented programming, high resolution Navier-Stokes (NS) solution methods, and unstructured mesh techniques, must all be pursued as possible paths toward higher fidelity mold filling simulations. A detailed exposition of these algorithmic developments is beyond the scope of this paper, hence we choose to focus here exclusively on algorithms for interface kinematics. These *interface tracking* algorithms are designed to model the movement of interfaces relative to a reference frame such as a fixed mesh. Current interface tracking algorithm choices are numerous, so is any one best suited for mold filling simulation? Although a clear winner is not (yet) apparent, pros and cons are given in the following brief review. Highlighted are those outstanding interface tracking algorithm issues that can hamper the reliable modeling of today's foundry mold filling processes.

Introduction And Perspective

Interfacial flows possessing multiple immiscible fluids bounded by topologically complex interfaces are ubiquitous in natural and industrial processes [1]. The active or passive filling of a mold in a casting process is an excellent example of a complicated interfacial flow. In a typical mold filling scenario, at least four "fluids", each separated from the other by a discernible interface, are present: the mold material; the mold cavity material, which is usually vacuum, air, or an inert gas; the molten liquid filling the mold cavity; and the solidified liquid, formed subsequent to mold fill provided adequate cooling has taken place.

Simulation of the interfacial flows found in mold filling via discrete numerical solution of appropriate partial differential equations is arguably the principal path toward a fundamental understanding of these flows. In modeling a mold filling process, the mold can be assumed to be relatively stationary, with movement brought about only by residual stress or distortion. Its bounding interface, although often topologically complex, can coincide with mesh surfaces partitioning the computational domain. The interface delineating the cavity gas and injected molten liquid, however, is not as easily modeled. During a typical mold filling process, this interface will possess topologies that are not only irregular but also *dynamic*, undergoing gross changes such as merging, tearing, and filamenting as a result of the flow and interface physics such as surface tension and phase change. The interface topology requirements facing an algorithm required to track the cavity gas/molten liquid interface are therefore formidable.

In the next section, we motivate the need for specialized algorithms designed to model cavity gas/molten liquid interface kinematics during mold filling. We next take a brief "tour" through most of the current interface tracking algorithms, then conclude with final thoughts.

Why Is Mold Filling Simulation So Difficult?

Consider first the physical phenomena likely to be present in a typical mold filling process: unsteady, incompressible (or slightly compressible) flow of multiple, immiscible fluids; interface kinematics and dynamics (surface tension, wetting effects); convective, diffusive, and radiative heat transfer; solidification of multi-component alloy systems having arbitrary phase diagrams; microstructural physics (nucleation, growth, kinetics); and material response effects (residual stress, distortion, shrinkage, plastic flow). While developing models to simulate each phenomenon above presents formidable, individual challenges, it is the simultaneous occurrence of these phenomena while a mold is being filled that presents *the* modeling challenge. During the filling of a mold, for example, one region of the casting might contain molten liquid free surface flow, another might be experiencing mushy zone porous flow, while still another contains solidified material that is generating internal stress as it cools. Each phenomenon takes place on disparate length scales. Much research remains before a fundamental understanding of these inherently nonlinear phenomena and their interplay with one another reaches a point to where first-principles models are realized.

Relative to the host of physical phenomena present during a mold fill process, can the motion and topology of the cavity gas/molten liquid interface really play a large role in determining the final characteristics of the cast part? In short, yes. This is especially true, however, if mold cavity geometries are complex and heat transfer rates are such that appreciable cooling (and/or solidification) of the molten liquid can take place during fill. If interface kinematics are not modeled correctly during mold fill, the resulting erroneous

interface topology and history could lead to flawed heat transfer predictions along the *mold* cavity boundary. Furthermore, since the distribution of (any) porosity and molten liquid momentum, energy, alloying species begins with the mold fill process, these could also be modeled incorrectly if the interface tracking algorithm is not a faithful representation. An accurate and reliable mold filling simulation therefore starts with a high fidelity interface tracking algorithm.

Interface Tracking Algorithms: A Tour

What specific capabilities might one design into an interface tracking algorithm targeted for mold filling simulation? The list is long. For high fidelity mold filling simulations, we desire an interface tracking algorithm that:

R1. is globally *and* locally mass conservative;
R2. maintains (at a minimum) second order temporal and spatial accuracy;
R3. maintains compact interface discontinuity width;
R4. is topologically robust;
R5. is amenable to three-dimensions on meshes of arbitrary element type/connectivity;
R6. can accommodate additional interfacial physics models;
R7. can track interfaces bounding more than two materials;
R8. is computationally efficient;
R9. can be implemented by novices (given ample documentation); and
R10. can be readily maintained, improved, and extended.

We do not require the algorithm be *implicit* in time, as this is unnecessary in most mold filling situations because resolution of the dynamics is desired. To date no one algorithm discussed in this paper adequately meets each and every design requirement itemized above.

Interface tracking algorithms generally fall into one of two methods categories, with each category containing several schemes (discussed in the next section):

tracking methods: moving-mesh, front tracking, boundary integral, particle schemes;
capturing methods: continuum advection, volume tracking, level set, phase field schemes.

A tracking method is Lagrangian in nature, whereby the position history of discrete points $\mathbf{x}_i$ lying on the interface are tracked for all time by integrating the evolution equation

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{V}_i\,, \tag{1}$$

where $\mathbf{V}_i$ is the velocity with which interface point x_i moves. For moving-mesh, front tracking, and boundary integral schemes, the points i correspond to the discrete points on the grid line representing the interface. For particle-based schemes, the points i correspond to individual particles (with known identity) lying along the interface. In capturing methods, the interface is not explicitly tracked, but rather "captured" using a characteristic function C that is the discontinuous Heaviside function in the limit of zero mesh spacing. For example, in the two-fluid case, C will take on constant values away from the interface:

$$C = \begin{cases} C_1 & \text{in fluid 1;} \\ C_2 & \text{in fluid 2;} \\ > C_1, < C_2 & \text{at the interface;} \end{cases} \tag{2}$$

where we assume $C_2 > C_1$. For finite mesh spacing, C is *not* perfectly discontinuous, hence the region with $C_1 < C < C_2$ has finite width, on the order of the mesh spacing. Since a point on the interface must remain there, the evolution equation for C is simply a statement of Lagrangian invariance:

$$\frac{DC}{Dt} = \frac{\partial C}{\partial t} + (\mathbf{V} \cdot \nabla) C = 0 , \tag{3}$$

where $\mathbf{V}$ is the velocity at the interface. Exact Lagrangian information is not retained in capturing methods (discarded instead for C), hence the interface location is not known exactly. Its position is defined as the transition region where $C_1 < C < C_2$. In capturing methods, knowledge of C allows one set of model equations to apply everywhere in the domain. Away from the interface, the equations reduce to the correct pure fluid equations, and within the interface, they contain appropriate discrete delta functions (formed from C) for interfacial terms [2,3].

Although differences between each scheme can be inferred from the discussions in the next section, some categorical tracking/capturing differences pervade. In capturing methods, the grid is usually fixed, hence topological robustness is inherent. Compact interface width, on the other hand, is difficult to insure because of the potential for numerical diffusion in discretization of the $(\mathbf{V} \cdot \nabla) C$ term in regions where C abruptly varies. In tracking methods, the interface is maintained as a discontinuity, yet 3-D topological robustness can be elusive.

Which schemes have been used for mold filling simulations? Several have been reported in the literature, yet the benchmark mold fill problem presented at this previous conference (MCWASP VII) showed that most brave enough to tackle this problem employed variations of volume tracking method [4]. Other methods are just (if not more) as viable, hence if this benchmark were to be repeated in the future, more entries using a wider variety of interface tracking algorithms would likely be submitted.

In the following short review, we comment on the ability of each algorithm to meet our requirement list. Some of our comments will undoubtedly reflect our own research experiences and tastes. Definitive, objective comments and conclusions should only be guided by detailed verification and validation studies [5]. For additional reviews from other general perspectives, the reader is encouraged to consult references [6–15].

Moving-Mesh Methods

Moving-mesh methods encompass all techniques that move the physical position of discrete grid points in the computational domain by integrating equation (1) forward in time. A moving-mesh method is Lagrangian if *every* point is moved, and mixed (Lagrangian-Eulerian) if grid points in a subset of the domain are moved. Mold filling simulations provide an excellent reason for using mixed methods [16], where the mold computational domain can be held stationary and the molten liquid is followed with a Lagrangian mesh [17,18]. With the mesh boundary-fitted to the physical domain, the system of equations may be transformed into logical space [15,19,20] or expressed as integrals over discrete control volumes [21]. Useful overviews can be found in classical [22] and more recent textbooks [12]. If the interfacial flow problem to be modeled has regular interface topologies, than a moving-mesh method can yield very accurate solutions. "Regular" topologies here refer to single-valued interface topologies that do not undergo tearing, stretching, or merging. Unfortunately regular topologies are not characteristic of those encountered in mold filling simulations.

Discrete control volumes (elements) in the computational domain encompass the same parcel of fluid in Lagrangian moving-mesh methods. If the velocity field has appreciable shear

or vorticity, then elements must distort and freely deform with the flow, yet they cannot because of their geometric limitations. Herein lies the principal drawback of these methods: element distortion leads to deterioration of solution accuracy and eventual termination of the simulation if element connectivity rules are violated. A solution to this problem is to remesh the domain, either by keeping the original mesh connectivity constant or allowing it to change. Lagrangian mesh methods that allow connectivity changes (including removal or creation of elements) are known as free Lagrange methods [23]. Both conventional Lagrangian [24,25] and free Lagrange [26] methods have been used incompressible interfacial flows. The principal problem with remeshing is the potential for global numerical diffusion and difficulty in demonstrating convergence. These issues must be quantified and overcome before moving-mesh methods can be competitive and viable for mold filling simulations.

Front Tracking Methods

Explicit front tracking has its roots in the original MAC method [27] and its extensions by Daly [28]. The interface is represented discretely by Lagrangian markers connected to form a front which lies within and moves through a stationary Eulerian mesh. As the front moves and deforms, interface points are added, deleted and reconnected as necessary. The interface can in principle undergo arbitrarily complex topology changes, provided the algorithm is capable of making logical topology decisions. Topological changes do not result from a set of localized interface physics models, instead resulting from algorithm intervention, via decisions such as whether or not to "cut" a front into pieces or to join two fronts into one. This algorithm as designed is *not* explicitly mass conservative, although conservation tends to be reasonably adhered to for topologically regular interfaces and high densities of marker particles (per unit interface length). Further details on the front tracking method can be found in [15,29–31].

Interface information is passed between the moving Lagrangian interface and the stationary Eulerian mesh using ideas borrowed from the immersed interface method [2,3]. With this technique, the sharp interface is approximated by a smooth distribution function that is used to distribute the sources at the interface over mesh points nearest the interface. The front is therefore given a finite (mesh size) thickness to provide stability and smoothness. Numerical diffusion is absent since this thickness remains constant for all time. The front tracking distribution function is the previously-mentioned function C, hence this portion of the algorithm is akin to capturing methods. Mass conservation is violated in front tracking methods because no single C contour will in general coincide with the interface formed by connecting the Lagrangian markers.

Front tracking methods have been successfully applied to a variety of interfacial flow problems: gas dynamics [29,30], incompressible flow with and without phase change [32–35], and microstructure evolution in alloy solidification [36,37]. It has also been recently demonstrated that front tracking methods can reasonably withstand 3-D topological challenges [34]. Key outstanding issues are whether the conservation properties and topological robustness are adequate for 3-D mold filling simulations. Implementation complexities could also continue to preclude its widespread acceptance and use.

Particle-Based Methods

Particle-based methods are characterized by the use of discrete "particles" to represent macroscopic fluid parcels [38]. The Lagrangian NS equations are integrated on globs of fluid (the "particles") having properties such as mass, momentum, and energy. Using a particle-based method for modeling interfacial flows is attractive because difficult nonlinear advection terms in the NS equations are simply modeled as particle motion, and, by knowing the identity and position of each particle, material interfaces are automatically

tracked. By using particle motion to approximate the advection terms, numerical diffusion across interfaces (where particles change identity) is virtually zero, hence interface widths remain compact. Particle-based methods can be roughly broken down into two principal categories: those that use particles in conjunction with a grid, namely the particle-in-cell (PIC) methods [39], and those that are "meshless" [40], such as the so-called smoothed particle hydrodynamics (SPH) methods [41, 42].

F. Harlow and coworkers invented the first particle-based method over forty years ago, the ingenious PIC method [43,44]. The PIC method (and its countless variations) then enjoyed explosive use and development over the next twenty years or so [39], where it became *the* method for modeling highly distorted interfacial flows. It has enjoyed a slow and steady resurgence since the mid 1980s, in large part due to innovative new developments and improvements [45–47]. Relative to Harlow's *classical* PIC method, modern *full* PIC methods force particles to carry all relevant fluid information (rather than only identity, position, and mass). This formulation was first adopted in the novel FLIP algorithm [45] for interfacial flows. The SPH method, first invented by J. Monaghan and coworkers over twenty years ago, differs from PIC methods in that an accompanying grid is *not* used. Like PIC methods, SPH methods are particularly well suited (and are generally designed) for high-speed compressible flows such as those found in astrophysical applications and shock dynamics.

Why have particle-based methods not been used more often (or at all) for simulating the free surface flows found in mold filling applications? Because particle-based methods can be prohibitively CPU and memory intensive, they have not had the ability to model low-speed (incompressible) flows until recently [48–51], they tend to be susceptible to subtle numerical instabilities, and they do not have adequate awareness and notoriety among researchers in the casting modeling community. If sustained attention is paid to these outstanding issues (e.g., the memory use issue is addressed in [5]), then many of these shortcomings could be alleviated, whereby particle-based methods could play a vital role in mold filling simulations. One powerful attraction to these methods is ease of implementation: they are typically no more complex to implement on 3-D unstructured meshes than on 2-D structured meshes.

Boundary Integral Methods

Methods of the boundary integral type [52, 53] can be highly accurate for modeling free surface flows, especially 2-D flows with relatively regular interface topologies. In this approach, the interface is explicitly tracked, as in moving-mesh or front tracking schemes, but the flow solution in the entire domain is deduced *solely* from information possessed by discrete points along the interface. For incompressible flows, the interface is characterized by a velocity potential, which is represented as a distribution of point dipoles. Boundary integral methods (BIM) were first used by Rosenhead [54] some sixty years ago to study vortex sheet roll-up, but it took another thirty years before extensions allowed the modeling of more general fluid interface problems [55]. Much evolution and enhancement has since occurred in the past two decades; see Yeung [6] and Hou [53] for reviews of earlier and more recent works, respectively. We include "vortex methods" in the class of boundary integral methods; see the excellent reviews by Leonard [56, 57]. These methods express the velocity of each discrete interface point as an integral relation that depends upon the vortex strength at that point.

The principal advantages gained by using boundary integral methods are the reduction of the flow problem by one dimension involving quantities of the interface only, and the potential for highly accurate solutions if the flow has topologically regular interfaces. Disadvan-

tages include difficulties in extending the method to 3-D (although it has been done [57]), sensitivity to numerical instabilities because the underlying problems are very singular, and the need for local "surgery" of the interface in the event of topological changes, much like the front tracking method. It is also not clear that BIM-based mold filling simulations could easily adapt to the complex mold cavity geometries. BIM-based free surface flow simulations, however, continue to generate impressive solutions for a wide variety of applications [58].

Continuum Advection Schemes

Continuum advection schemes refer to traditional methods for obtaining discrete numerical solutions to equation (3). This equation is perhaps the simplest form of a hyperbolic equation, hence the countless references and textbooks available on hyperbolic equation solution techniques can be drawn upon [59–62]. We classify these schemes as continuum advection schemes since they are usually designed upon the premise that C in equation (3) is smoothly varying. A simple yet highly diffusive example is a first-order upwind "donor cell" scheme. More accurate examples are the higher-order monotonic van Leer [63], PPM [64], and TVD [65] schemes. Techniques similar to these schemes should be used in mold filling simulations for solutions to (3) if C represents a passive scalar, energy, solute species, momentum, or density away from interfaces.

Continuum advection schemes have been employed as interface tracking methods in mold filling simulations [66, 67]. Asking such schemes to track interfaces, however, forces them to generate solutions to (3) when C is a discontinuous. This is problematic because these schemes are not designed for this purpose. The source of this problem lies in the algebraic treatment of the $(\mathbf{V} \cdot \nabla) C$ term; even with higher-order approximations, unacceptable broadening of the region where C varies occurs. For example, a compressive, fourth-order PPM method was still found to unacceptably diffuse the interface [5]. Why? Because discontinuities in C can only remain so after solutions to (3) result from *geometric* approximations to $(\mathbf{V} \cdot \nabla) C$; continuum advection schemes approximate this term *algebraically*, i.e, by taking spatial differences in C directly across the interface. Higher-order approximations to these differences can obviously mitigate this numerical diffusion, yet not nearly well enough. Studies in addition to our own [5] found that interface widths can broaden to many cells (4–8) even when higher-order methods are used [68]. This is not satisfactory for an interface tracking method tasked to perform 3-D mold filling simulations.

One interesting approach to the interface diffusion problem is to transform the discontinuous C function into another *smooth* function $\tilde{C}$, solve (3) with $\tilde{C}$, then transform $\tilde{C}$ back to C. In this way, the continuum advection scheme can generate solutions for which it was designed. This idea, proposed in [69] and elsewhere, is also the basic premise of the level set method [13].

Volume Tracking Methods

As stated previously, most of the MCWASP VII benchmark mold fill problem submissions employed volume tracking methods. One should not infer from this, however, that this algorithm is the best choice for tracking interfaces in mold filling simulations. It remains the most popular and widely used tracking method for mold filling, but this could be due to reasons other than performance: it is relatively easy to implement (at least in its crude forms), it is an older, well-documented algorithm, it is topologically robust, and its basis in volume fractions lends itself well to incorporation of other physics. Volume tracking methods differ from continuum advection schemes in one principal way: the $(\mathbf{V} \cdot \nabla) C$ in equation (3) is approximated geometrically, based upon knowledge of a (nonunique) "reconstructed" interface position.

Volume tracking methods originated in the early 1970s, when three methods were introduced within a short period of time: DeBar's method [70], Hirt and Hichols' VOF method [71, 72], and Noh and Woodward's SLIC method [73]. See [74, 75] for historical perspectives and accounts of the chronological developments that have taken place over the last three decades. In short, *substantial* evolutionary development and improvement has taken place, rendering most of these original methods virtually obsolete [76]. Their spatial and temporal first order accuracy is just not competitive nor adequate for modern simulations.

Without certain key developments occurring over the last decade, volume tracking methods would not have remained competitive with newer, impressive methods such as front tracking and level set techniques. Such developments include spatially second order, linearity-preserving interface geometry reconstructions, multi-dimensional time integration schemes [74, 75, 77], and extensions to 3-D unstructured meshes [78]. Perhaps the most important trend is the movement of volume tracking algorithms away from heuristic "case-by-case" logic [79] toward a mathematically formal algorithm based upon geometric primitives. This trend should facilitate propagation of improvements to older implementations, reduce the likeliness of any two volume tracking algorithms generating vastly different solutions, reduce implementation difficulties, and make it easier for future researchers to find weaknesses and devise improvements.

Given these recent developments, modern volume tracking methods are competitive with other interface tracking methods [5, 80]. They will remain viable in the future, but several outstanding issues should be resolved. Some of these include quantification and alleviation of numerical surface tension, improved schemes for time integration and interface normal approximations, and improved efficiency and ease of implementation on arbitrary 3-D meshes[1].

Level Set Methods

Since introduction of the level set method by Osher and Sethian [81] only a decade ago, its use has exploded, evidenced by the recent textbook and references therein [13]. To date, the level set method has been used to model interfacial phenomena in the fields of material science, fluid mechanics, image enhancement, computer vision, and grid generation, to name a few. The mathematical formalism and rigor grounded in the level set method has helped to attract leading numerical analysts and mathematicians, resulting in its evolution, widespread promotion and use, and increasing range of applicability.

The basic premise of the level set method is to embed the propagating interface $\Gamma(t)$ as the zero level set of a higher dimensional function ϕ, defined as $\phi(\mathbf{x}, t = 0) = \pm d$, where d is the distance from $\mathbf{x}$ to $\Gamma(t = 0)$, chosen to be positive(negative) if $\mathbf{x}$ is outside(inside) the initial $\Gamma(t = 0)$. If the zero level set coincides with the initial interface, i.e., $\Gamma(t = 0) = \phi(\mathbf{x}, t = 0) = 0$, then a dynamical equation for $\phi(\mathbf{x}, t)$ that contains the embedded motion for $\Gamma(t)$ as the level set $\phi = 0$ can be derived [81]:

$$\frac{\partial \phi}{\partial t} + F|\nabla \phi| = 0\,, \tag{4}$$

where F is the speed of the interface Γ in the outward normal direction. In general, F is the sum of any applied interface propagation speed, a curvature-dependent speed, and the flow velocity normal to interface, i.e., $\mathbf{V} \cdot \hat{\mathbf{n}}$, where $\hat{\mathbf{n}} = \nabla\phi/|\nabla\phi|$ is the unit interface normal. For certain forms of F this equation takes on a standard Hamilton-Jacobi form, but for the interfacial flows encountered in mold filling, F is only $\mathbf{V} \cdot \hat{\mathbf{n}}$, hence (4) equivalent to (3).

[1] Further details can be found in the papers available at `www.lanl.gov/home/Telluride`.

Level set methods, then, propagate interfaces by integrating the same basic scalar evolution equation as other capturing methods. The *difference*, however, is that the scalar function ϕ in level set methods is *not* some discrete representation of a Heaviside (H) function, but rather a smoothly varying *distance* function. Herein lies a key advantage: highly accurate numerical solutions to equation (3) are possible (e.g., using the continuum advection schemes previously mentioned) since ϕ is smoothly varying. Many such schemes are readily available and easily implemented [61]. Herein also lies a key disadvantage: since H is not directly integrated forward in time, it must be reconstructed each time step from the distance function ϕ, $\tilde{H}(\phi)$ [58]. This procedure will not be mass conservative [5] since densities are defined from $\tilde{H}(\phi)$ rather than an H that is directly integrated forward in time according to strict mass conservation.

Rigorous mass conservation is elusive in the level set method because ϕ does not necessarily remain a distance function after solutions to (4) are obtained. This is especially true if the interface has undergone large topological changes. So-called "reinitialization" algorithms have therefore been devised [58, 82] to insure ϕ remains a distance function, i.e., satisfying $|\nabla\phi| = 1$. Reinitialization can (and in general will) move the zero level set position; this violates mass conservation. More accurate information is needed about the front position so that movement of the zero level set does not occur during reinitialization. A global mass conservation constraint [58], acting like a Lagrange multiplier, has improved conservation properties of the reinitialization, but *local* constraints are still needed. Without local constraints, the zero level set might still move as much as a cell width during reinitialization, which artificially creates mass locally in some cells and destroys it in others. This is not acceptable performance for an interface tracking algorithm used in mold filling simulations. If local, *geometric* information is used to fix the zero level set position during reinitialization, the algorithm is likely to have many similarities to volume tracking algorithms [83].

Phase Field Formulations

Phase field formulations have been applied to crystal growth problems and Hele-Shaw flows over the past decade [84–87], but only recently have they shown promise for NS flows [88–90]. Phase field models, like other Eulerian capturing methods, model interfacial forces as continuum forces by smoothing interface discontinuities and forces over thin but numerically resolvable layers. This smoothing allows conventional numerical approximations of interface kinematics on fixed grids. The phase field method also provides a continuum surface tension model [91]) that is energetically and thermodynamically consistent [90]. Just recently a phase field model for dendritic solidification in the presence of melt convection (incompressible NS flow) was developed [92]. Phase field formulations are indeed showing promise and the ability to provide a powerful vehicle for the direct numerical simulation of interfacial phenomena.

In the case of free surface flows, the starting point is the van der Waal hypothesis, in which the interfacial energy density depends upon both ϕ (the phase field) and gradients in ϕ. Cahn and Hilliard [93] extended this hypothesis to dynamical situations by approximating interfacial diffusion fluxes as being proportional to chemical potential gradients. Jacqmin [90] recently extended the Cahn-Hilliard equation to allow for the presence of flow. Equation (3) gives Jacqmin's evolution equation for ϕ (where $C = \phi$), except that the RHS side is the Laplacian of the chemical potential rather than zero. This term is quite interesting; it can be diffusive (positive) or anti-diffusive (negative), helping to regularize the interface width (not too diffuse or compact). Rather than relying on special numerical techniques in tracking algorithms to keep interface widths regular, physical mechanisms in the phase field method do the work. Simple central difference expressions for $(\mathbf{V} \cdot \nabla) C$

were found to be adequate in most cases [90].

The presence of the (anti)diffusive term on the RHS of equation (3) is problematic, however: at least three cells are needed through the interface so that the Laplacian can be properly discretized, otherwise the interface will "stick" to the mesh [90]. Current approaches aimed to alleviate this problem are to adaptively refine the interface region; this (or some other solution) will be required for phase field methods to be viable in 3-D. Perhaps the physical principles embodied in these formulations can be combined with the numerical techniques in tracking methods to yield an improved, unified approach. Phase field formulations for free surface flows are new and exciting, and deserve further attention and exploration.

Final Thoughts

The wide variety of interface tracking algorithms reviewed in this paper are evidence for the many options available for modeling interfacial flows. Keeping in mind that each algorithm has its own unique strengths and weaknesses, which algorithm possesses strengths best suited for mold filling simulation? Objective answers will follow only if additional systematic studies are carried out to: (1) execute controlled tests with defined flow fields and known interface topology solutions [5], (2) perform interfacial flow simulations using varied interface tracking algorithms and the same NS flow solver [77, 80], and (3) help validate mold filling simulation results against experimental data. Too few of these studies have been performed, hence there is not (yet) a clear interface tracking algorithm "winner". Our own experiences with volume tracking, front tracking, and particle-based methods have led us to believe that these methods have been and will continue to be useful. The more recent level set and phase field methods have also exhibited impressive capabilities. Since one algorithm's strength is often another's weakness, a very real (*and welcome*) possibility is the convergence of many of these methods toward one unified method. Not to be overlooked is the possibility that some new, robust *and* accurate 3-D method is yet to be discovered.

References

[1] Jens Eggers. *Reviews of Modern Physics*, 69:865–929, 1997.

[2] C. S. Peskin. *Journal of Computational Physics*, 25:220–252, 1977.

[3] R. J. Leveque and Z. Li. *SIAM Journal on Numerical Analysis*, 31:1019–1044, 1994.

[4] B. Sirrell, M. Holliday, and J. Campbell. In M. Cross and J. Campbell, editors, *Modeling of Casting, Welding, and Advanced Solidification Processes VII*, pages 915–933, New York, 1996. TMS Publishers.

[5] W. J. Rider and D. B. Kothe. Technical Report AIAA 95–1717, AIAA, 1995. Presented at the 12th AIAA CFD Conference.

[6] R. W. Yeung. *Annual Reviews in Fluid Mechanics*, 14:395–442, 1982.

[7] J. Crank. *Free and Moving Boundary Problems.* Oxford University Press, 1984.

[8] J. M. Hyman. *Physica D*, 12:396–407, 1984.

[9] N. Shamsundar and E. Rooz. In W. J. Minkowycz, E. M. Sparrow, G. E. Schneider, and R. H. Pletcher, editors, *Handbook of Numerical Heat Transfer*, pages 747–786, New York, 1988. Wiley.

[10] H. P. Wang and H. S. Lee. In C. L. Tucker, editor, *Fundamentals of Computer Modeling for Polymer Processing*, pages 369–401, New York, 1989. Hanser Publishers.

[11] J. M. Floryan and H. Rasmussen. *Applied Mechanics Reviews*, 42:323–340, 1989.

[12] E. S. Oran and J. P. Boris. *Numerical Simulation of Reactive Flow.* Elsivier, 1987.

[13] J. A. Sethian. *Level Set Methods.* Cambridge University Press, 1996.

[14] W. Tsai and D. K. P. Yue. *Annual Reviews in Fluid Mechanics*, 28:249–278, 1996.

[15] W. Shyy, H. Udaykumar, M. Rao, and R. Smith. *Computational Fluid Dynamics with Moving Boundaries.* Taylor and Francis, 1996.

[16] W. F. Noh. *Methods in Computational Physics*, 3:117–179, 1964.

[17] F. Muttin, T. Coupez, M. Bellet, and J. L. C. Chenot. *International Journal for Numerical Methods in Engineering*, 36:2001–2015, 1993.
[18] R. W. Lewis, S. E. Navti, and C. Taylor. *International Journal for Numerical Methods in Fluids*, 25:931–952, 1997.
[19] J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin. *Numerical Grid Generation.* Elsevier Science, New York, 1985.
[20] C. A. J. Fletcher. *Computational Techniques for Fluid Dynamics: Volume II.* Springer-Verlag, 1988.
[21] M. Vinokur. *Journal of Computational Physics*, 81:1–52, 1989.
[22] R. D. Richtmyer and K. W. Morton. *Difference Methods for Initial Value Problems.* Wiley-Interscience, 1967.
[23] M. J. Fritts, W. P. Crowley, and H. E. Trease. *The Free Lagrange Method.* Springer-Verlag, New York, 1985. Lecture Notes in Physics, Volume 238.
[24] C. W. Hirt, J. L. Cook, and T. D. Butler. *Journal of Computational Physics*, 5:103–124, 1970.
[25] P. Bach and O. Hassager. *Journal of Fluid Mechanics*, 152:173–210, 1985.
[26] M. J. Fritts and J. P. Boris. *Journal of Computational Physics*, 31:173–215, 1979.
[27] F. H. Harlow and J. E. Welch. *Physics of Fluids*, 9:842–851, 1966.
[28] B. J. Daly. *Physics of Fluids*, 10:297–307, 1967.
[29] J. Glimm and O. A. McBryan. *Advances in Applied Mathematics*, 6:422–435, 1985.
[30] I-L. Chern, J. Glimm, O. McBryan, B. Plohr, and S. Yaniv. *Journal of Computational Physics*, 62:83–110, 1986.
[31] W. Shyy. *Computational Modeling for Fluid Flow and Interfacial Transport.* Elsevier Science, 1994.
[32] D. Juric and G. Tryggvason. Technical Report LA-UR-97-1145, Los Alamos National Laboratory, 1997. Accepted for Publication in the International Journal of Multiphase Flow.
[33] D. Juric and G. Tryggvason. *Heat Transfer in Microgravity Systems*, 332:33–44, 1996.
[34] S. O. Unverdi and G. Tryggvason. *Journal of Computational Physics*, 100:25–37, 1992.
[35] S. O. Unverdi and G. Tryggvason. *Physica D*, 60:70–83, 1992.
[36] D. Juric and G. Tryggvason. *Journal of Computational Physics*, 123:127–148, 1996.
[37] D. Juric. In *Modeling of Casting, Welding, and Advanced Solidification Processes VIII*, New York, 1998. TMS Publishers. These proceedings.
[38] J. J. Monaghan. *Computer Physics Reports*, 3:71–124, 1985.
[39] F. H. Harlow. *Computer Physics Communications*, 48:1–11, 1988.
[40] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. *Computer Methods in Applied Mechanics and Engineering*, 139:3–47, 1996.
[41] R. A. Gingold and J. J. Monaghan. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977.
[42] J. J. Monaghan. *Annual Reviews in Astronomy and Astrophysics*, 30:543–574, 1992.
[43] F. H. Harlow and M. W. Evans. Technical Report LAMS-1956, Los Alamos National Laboratory, 1955.
[44] F. H. Harlow. *Journal of the Association for Computing Machinery*, 4:137–142, 1957.
[45] J. U. Brackbill and H. M. Ruppel. *Journal of Computational Physics*, 65:314–343, 1985.
[46] J. U. Brackbill, D. B. Kothe, and H. M. Ruppel. *Computer Physics Communications*, 48:25–38, 1988.
[47] D. B. Kothe, J. U. Brackbill, and C. K. Choi. *Physics of Fluids B*, 2:1898–1906, 1990.
[48] D. B. Kothe and J. U. Brackbill. FLIP-INC: A Particle-in-Cell method for incompressible flows, 1992. unpublished manuscript.
[49] J. J. Monaghan. *Journal of Computational Physics*, 110:399–406, 1994.
[50] J. P. Morris, P. J. Fox, and Y. Zhu. *Journal of Computational Physics*, 136:214–226, 1997.
[51] Joseph P. Morris. *International Journal for Numerical Methods in Fluids*, 1997. Submitted.
[52] C. Pozrikidis. *Boundary Integral and Singularity Methods for Linearized Viscous Flow.* Cambridge University Press, 1992.
[53] T.Y. Hou. *Acta Numerica*, pages 335–415, 1995.
[54] L. Rosenhead. *Proceedings of the Royal Society of London, Series A*, 134:170–192, 1932.
[55] G. Birkhoff. In *Symposium on Applied Mathematics*, volume 13, pages 55–76, Providence, RI, 1962. American Mathematical Society. Volume XIII.
[56] A. Leonard. *Journal of Computational Physics*, 37:289–335, 1980.
[57] A. Leonard. *Annual Reviews in Fluid Mechanics*, 17:523–559, 1985.

[58] M. Sussman and P. Smereka. *Journal of Fluid Mechanics*, 341:269–294, 1997.
[59] P. Woodward and P. Colella. *Journal of Computational Physics*, 54:115–173, 1984.
[60] S. T. Zalesak. In R. Vichnevetsky and R. S. Stepleman, editors, *Advances in Computer Methods for Partial Differential Equations*, volume 6, pages 15–22. IMACS, 1987.
[61] R. J. Leveque. *Numerical Methods for Conservation Laws*. Birkhäuser, 1990.
[62] J. H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer-Verlag, 1996.
[63] B. van Leer. *Journal of Computational Physics*, 23:276–299, 1977.
[64] P. Colella and P. Woodward. *Journal of Computational Physics*, 54:174–201, 1984.
[65] P. K. Sweby. In B. Engquist, editor, *Lectures in Applied Mathematics*, volume 22, pages 289–309, 1985.
[66] K. S. Chan, K. A. Pericleous, and M. Cross. *Applied Mathematical Modelling*, 15:624–631, 1991.
[67] K. A. Pericleous, K. S. Chan, and M. Cross. *Numerical Heat Transfer, Part B*, 27:487–507, 1995.
[68] B. Fryxell, E. Müller, and D. Arnett. *Astrophysical Journal*, 367:619–634, 1991.
[69] T. Yabe and F. Xiao. *Computers in Mathematics Applications*, 29:15–25, 1995.
[70] R. DeBar. Technical Report UCIR–760, Lawrence Livermore National Laboratory, 1974.
[71] B. D. Nichols and C. W. Hirt. Technical Report LA–UR–75–1932, Los Alamos National Laboratory, 1975.
[72] C. W. Hirt and B. D. Nichols. *Journal of Computational Physics*, 39:201–225, 1981.
[73] W. F. Noh and P. R. Woodward. In A. I. van de Vooren and P. J. Zandbergen, editors, *Lecture Notes in Physics 59*, pages 330–340, 1976.
[74] D. B. Kothe, W. J. Rider, S. J. Mosso, J. S. Brock, and J. I. Hochstein. Technical Report AIAA 96–0859, AIAA, 1996. Presented at the 34rd Aerospace Sciences Meeting and Exhibit.
[75] W. J. Rider and D. B. Kothe. Technical Report LA–UR–96–2375, Los Alamos National Laboratory, 1998. Accepted for publication in the Journal of Computational Physics.
[76] D. B. Kothe and W. J. Rider. Technical Report LA–UR–3384, Los Alamos National Laboratory, 1994.
[77] M. Rudman. *International Journal for Numerical Methods in Fluids*, 24:671–691, 1997.
[78] A. V. Reddy, D. B. Kothe, C. Beckermann, R. C. Ferrell, and K. L. Lam. In *Solidification Processing 1997: Proceedings of the Fourth Decennial International Conference on Solidification Processing*, pages 83–87, The University of Sheffield, Sheffield, UK, July 7–10, 1997.
[79] S.-O. Kim and H. C. No. *International Journal for Numerical Methods in Fluids*, 26:79–100, 1998.
[80] W. J. Rider, D. B. Kothe, S. J. Mosso, J. H. Cerruti, and J. I. Hochstein. Technical Report AIAA 95–0699, AIAA, 1995. Presented at the 33rd Aerospace Sciences Meeting and Exhibit.
[81] S. Osher and J. A. Sethian. *Journal of Computational Physics*, 79:12–49, 1988.
[82] M. Sussman, P. Smereka, and S. Osher. *Journal of Computational Physics*, 114:146–159, 1994.
[83] A. A. Bourlioux. In H. A. Dwyer, editor, *Proceedings of the Sixth International Symposium on Computational Fluid Dynamics*, pages 15–22, Lake Tahoe, NV, 1995.
[84] G. Caginalp. *Physical Review A*, 39:5887–5896, 1989.
[85] R. Kobayashi. *Physica D*, 63:410–423, 1993.
[86] S. L. Wang, R. F. Sekerka, A. A. Wheeler, B. T. Murray, S. R. Coriell, and G. B McFadden. *Physica D*, 69:189–200, 1993.
[87] A. A. Wheeler, B. T. Murray, and R. J. Schaefer. *Physica D*, 66:243–262, 1993.
[88] L. K. Antanovskii. *Physics of Fluids*, 7:747–753, 1995.
[89] D. Jacqmin. Technical Report 96–0858, AIAA, 1996. Presented at the 34th Aerospace Sciences Meeting.
[90] D. Jacqmin. Technical report, NASA Lewis Research Center, 1997. Submitted to the Journal of Computational Physics.
[91] J. U. Brackbill, D. B. Kothe, and C. Zemach. *Journal of Computational Physics*, 100:335–354, 1992.
[92] X. Tong, C. Beckermann, and A. Karma. In B. Thomas and C. Beckermann, editors, *Modeling of Casting, Welding, and Advanced Solidification Processes VIII*, New York, 1998. TMS Publishers. These proceedings.
[93] J. W. Cahn and J. E. Hilliard. *Journal of Chemical Physics*, 31:688–699, 1959.